Off-Policy and On-Policy Deep Reinforcement Learning in a Pygame-Based Squash Environment: A Fixed-Budget Comparison of Deep Q-Network and Proximal Policy Optimization

Altan Ulaş Zöhre
Izmir Institute of Technology,
Department of Computer Engineering
Izmir, Türkiye
auzohre@gmail.com

Deniz Sakaroğlu
Izmir Institute of Technology,
Department of Computer Engineering
Izmir, Türkiye
denisakaroglu@gmail.com

***Abstract*— This study compares two major paradigms in deep reinforcement learning under the same interaction budget: an off-policy value-based method, Deep Q-Network, and an on-policy policy-gradient method, Proximal Policy Optimization. To enable a controlled and reproducible benchmark, we developed a custom squash/paddle-ball environment in Pygame and exposed it through a Gymnasium-compatible interface. The environment uses a discrete three-action control scheme (stay, move left, move right) and a five-dimensional normalized observation vector (paddle position, ball position, and velocity components). Both algorithms were trained with two multilayer perceptron capacities (128×128 and 256×256 hidden layers) under two fixed training budgets (50,000 and 100,000 environment steps). The learned policies were evaluated using deterministic action selection over N=50 evaluation episodes, and we report score as the primary task-success metric, with episodic return and episode length as supportive measures. Results indicate that the larger Proximal Policy Optimization configuration achieves the highest and most stable scores at both budgets, while the smaller Proximal Policy Optimization model improves substantially when the training budget is increased. In contrast, the tuned larger Deep Q-Network configuration remains significantly weaker across both budgets, highlighting the sensitivity of value-based learning to exploration and stability choices under limited interaction budgets.**

Keywords*— *Reinforcement Learning, Deep Reinforcement Learning, DQN, PPO, Simulation-Based Learning, Agent Evaluation

I. Introduction

Deep reinforcement learning enables an agent to learn a policy that maximizes long-term reward through interaction with an environment. In practice, however, deep reinforcement learning methods differ fundamentally in how they collect and reuse experience. **Off-policy** methods can reuse past transitions and may achieve higher sample efficiency, whereas **on-policy** methods update the policy using trajectories generated by the current policy, often leading to more stable optimization behavior. This distinction motivates comparisons that control not only for computational effort, but also for the number of environment interactions available for learning.

The goal of this work is to provide a fair comparison between an off-policy value-based method, **Deep Q-Network**, and an on-policy policy-gradient method, **Proximal Policy Optimization**, under the same environment and the same interaction budget. To enable controlled experimentation, we develop a custom squash/paddle-ball environment in Pygame and expose it through a Gymnasium-compatible interface. We evaluate both algorithms under two network capacities (128×128 and 256×256 multilayer perceptrons) and two training budgets (50,000 and 100,000 environment steps). Performance is measured using deterministic action selection over **N=50** evaluation episodes; **score** is used as the primary task-success metric, while **episodic return** and **episode length** are reported as supportive metrics.

This study addresses the following research questions:

i. Under a fixed interaction budget, how do Deep Q-Network and Proximal Policy Optimization differ in task success (score) and learning dynamics?
ii. When the interaction budget is increased, which algorithm–capacity combinations benefit the most, indicating higher sample efficiency?
iii. In the presence of reward shaping, how should score and episodic return be interpreted relative to each other?

Our main contributions are: (i) implementing a Pygame-based squash environment with a Gymnasium-compatible interface, (ii) conducting a fair 2×2×2 factorial evaluation across algorithm class, network capacity, and training budget, and (iii) systematically logging training and evaluation metrics to enable reproducible numerical reporting and visualization.

II. Literature Review

Reinforcement learning (RL) algorithms often differ substantially in data efficiency and training stability, making controlled comparisons important under limited interaction budgets. Yu et al. challenge the common assumption that on-policy methods are necessarily sample-inefficient by demonstrating that PPO can be surprisingly effective in practice when appropriate training practices are applied, even in complex multi-agent settings [1]. This perspective provides motivation for analyzing PPO performance under fixed environment-step budgets, where stability and data efficiency become central evaluation factors in our study.

Several works provide direct empirical evidence for differences between value-based and policy-based methods. Kozlica et al. present a head-to-head comparison of DQN and PPO within a unified simulation environment, reporting that PPO outperforms DQN across multiple task-success and episode-level metrics [2]. This study directly supports the relevance of evaluating DQN and PPO under the same experimental constraints and strengthens the comparative framing adopted in our work.

Arcade-style control tasks further highlight known limitations of vanilla DQN baselines. Chen et al. study reinforcement learning in Breakout and compare DQN against Double DQN (DDQN), emphasizing that addressing overestimation bias can improve both performance and learning stability [3]. Such findings align with our observation that DQN performance can be sensitive to training dynamics, and they motivate interpreting vanilla DQN results cautiously as well as considering stronger off-policy baselines (e.g., DDQN/Rainbow variants) in future work.

More closely related evidence comes from Breakout-specific comparative evaluations that include both DQN and PPO. de la Fuente and Vidal Guerra compare DQN, PPO, and A2C on the Breakout environment and report that performance is influenced not only by algorithm choice but also by hyperparameter settings such as learning rate and discount factor [4]. This observation supports a key point in our study: under fixed interaction budgets, reported outcomes can be highly sensitive to training configuration, making stability-focused evaluation and consistent protocols essential.

In addition to game-based studies, comparative analyses across different environment types indicate that algorithm advantages may depend on task characteristics. Tan compares PPO and DQN on CartPole (discrete control) and CarRacing (continuous control), evaluating convergence speed, stability, sample efficiency, and computational cost [5]. The study reports environment-dependent trade-offs and underscores the importance of reporting multiple complementary metrics rather than relying on a single scalar measure. This directly supports our evaluation design, which reports score as the primary task-success metric alongside episodic return and episode length to reflect the effects of reward shaping.

Finally, data-constrained learning has been recognized as a major challenge in deep RL. Prudencio et al. survey offline RL and discuss how performance depends strongly on data coverage and distributional properties, reinforcing the broader point that evaluation protocols must explicitly account for interaction constraints [6]. Complementary to this, Barreto et al. propose generalized policy updates (GPE/GPI) to accelerate learning by transferring knowledge across tasks, explicitly targeting improved data efficiency when interaction is expensive [7]. Together, these works strengthen the methodological motivation of our fixed interaction-budget setting: when environment interaction is limited, fair comparisons should emphasize sample efficiency and stability in addition to final performance.

III. Methodology

A. Environment

We design a custom squash/paddle-ball environment in Pygame and expose it through a Gymnasium-compatible interface. The task is modeled as an episodic Markov decision process in which the agent controls a horizontal paddle to keep the ball in play and maximize the number of successful returns.

The agent has no prior knowledge of the environment dynamics or reward function and learns a policy solely through trial-and-error interaction.

1. Observation space

Each time step provides a five-dimensional continuous observation vector consisting of: (i) normalized paddle horizontal position, (ii) normalized ball horizontal position, (iii) normalized ball vertical position, and (iv–v) scaled horizontal and vertical ball velocity components. Positions are normalized by the screen dimensions, and velocities are scaled to keep feature magnitudes bounded and numerically stable during learning.

2. Action space

The action space is discrete with three actions: stay, move left, and move right. The paddle position is updated with a fixed step size and clipped to remain within the screen boundaries.

3. Dynamics and termination.

Transitions are deterministic. The ball moves according to its velocity and bounces off the side walls and the top boundary by inverting the corresponding velocity component. A paddle collision near the bottom region inverts the vertical velocity and increments the game score; to gradually increase difficulty, the ball speed is mildly scaled up after successful hits (up to a cap). An episode terminates when the ball crosses

the bottom boundary (a miss). No explicit time-limit truncation is used.

4. Reward design

The reward includes (i) a large positive reward for successful paddle hits and a large negative terminal penalty for misses, and (ii) dense shaping based on the horizontal distance between the paddle center and the ball position (positive when the distance decreases and negative otherwise). A small per-step living reward encourages survival. Because shaping affects the accumulated return, we treat score as the primary task-success metric during evaluation. Fig. 1 provides a snapshot of the custom Pygame environment and the side-by-side monitoring dashboard used for qualitative inspection.

Fig. 1. Real-time Pygame dashboard for the custom Squash task, showing side-by-side rollouts for Deep Q-Network and Proximal Policy Optimization agents (small vs. large multilayer perceptrons) with live score statistics.

B. Learning Algorithms

We compare an off-policy value-based method (Deep Q-Network) with an on-policy policy-gradient method (Proximal Policy Optimization) under identical environment interaction budgets.

Deep Q-Network. Deep Q-Network learns an approximation of the action-value function using a neural network. It uses a replay buffer to reuse stored transitions via mini-batch sampling and a target network to stabilize bootstrapped targets. Exploration follows an epsilon-greedy strategy that decays over training. We evaluate a smaller configuration and a larger tuned configuration, differing in network capacity and key hyperparameters such as replay capacity, learning-start threshold, and target update frequency.

C. Network Architectures

To study the effect of model capacity, each algorithm is trained using two multilayer perceptron architectures with two hidden layers: a small network (128×128 units) and a large network (256×256 units). The observation vector is used as input and the networks output either action values (for Deep Q-Network) or policy/value predictions (for Proximal Policy Optimization). We follow common practice by using Rectified Linear Unit activations for Deep Q-Network models and hyperbolic tangent activations for Proximal Policy Optimization models.

D. Training and Logging

Fairness is enforced by controlling the number of environment interactions rather than wall-clock time. Each configuration is trained under a fixed budget of either 50,000 or 100,000 environment steps. Episode-level training metrics (episodic return and episode length) are recorded via monitoring utilities and a custom callback, and exported to comma-separated value files to support learning-curve visualization and reproducibility.

E. Fixed Evaluation Protocol

After training, each model is evaluated for N = 50 episodes using deterministic action selection to remove exploration noise from performance measurement. For each episode, we record score (successful returns), episodic return (including shaping and living rewards), and episode length. We report both episode-level traces and summary statistics (mean, standard deviation, median, and best score).

IV. Experimental Work

This section reports the numerical results obtained under fixed interaction budgets and discusses the effect of algorithm class, network capacity, and training budget on task performance. All models are evaluated using deterministic action selection over N = 50 evaluation episodes. We treat score (successful returns) as the primary task-success metric, while episodic return and episode length are reported as supportive measures.

A. Results After 50,000 Environment Steps

TABLE I. summarizes performance after training for 50,000 environment steps. The larger Proximal Policy Optimization configuration achieves the highest mean score (28.68 ± 4.32) and remains comparatively stable across evaluation episodes. The smaller Proximal Policy Optimization configuration produces a lower mean score (13.70 ± 7.16) and shows higher variability, indicating less consistent control under this limited budget.

On the off-policy side, the smaller Deep Q-Network attains a moderate mean score (23.28 ± 2.91), while the tuned larger Deep Q-Network remains in a low-score regime (2.20 ± 0.98). This outcome suggests that increasing capacity alone does not guarantee improved performance for value-based learning and that stability and exploration choices can dominate results under limited interaction budgets. Per-episode evaluation scores over 50 episodes are shown in Fig. 2, highlighting stability and variability across methods.

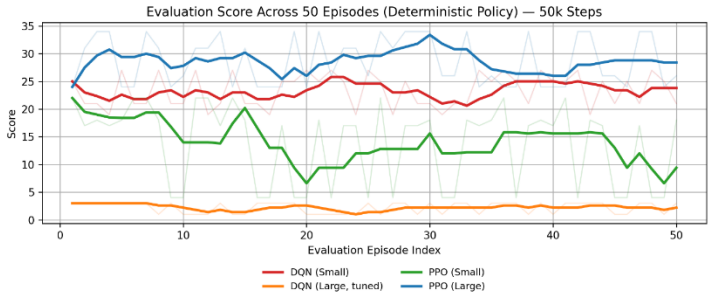

Fig. 2. Evaluation score across 50 episodes after 50k training steps (deterministic policy).

B. Results After 100,000 Environment Steps

TABLE II. reports results after training for 100,000 environment steps. The larger Proximal Policy Optimization model maintains the strongest mean score (28.64 ± 3.65),

remaining essentially unchanged relative to the 50,000-step condition. This indicates that the larger on-policy model reaches a strong performance level early and exhibits near-saturated task success within the tested budget range.

The most pronounced improvement is observed for the smaller Proximal Policy Optimization configuration, whose mean score increases from 13.70 to 26.80 as the budget doubles. In contrast, the smaller Deep Q-Network shows only a limited improvement (23.28 → 25.18). The tuned larger Deep Q-Network does not benefit from the additional interaction data and remains substantially weaker (1.30 ± 1.22), suggesting sensitivity to off-policy stability and exploration choices in this environment. Fig. 3 reports the corresponding per-episode evaluation scores after 100k training steps, where PPO (Small) exhibits a clear improvement.

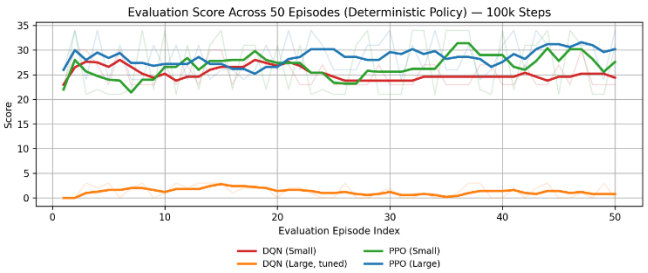

Fig. 3. Evaluation score across 50 episodes after 100k training steps (deterministic policy).

TABLE I. *Evaluation summary after 50,000 training steps (N = 50).*

Model	Score (mean ± std)	Median score	Best score	Return (mean ± std)	Mean length
Deep Q-Network (128×128)	23.28 ± 2.91	25.0	27	1339.46 ± 70.52	3463
Deep Q-Network (256×256, tuned)	2.20 ± 0.98	3.0	3	198.65 ± 73.03	680
Proximal Policy Optimization (128×128)	13.70 ± 7.16	17.0	22	495.70 ± 218.16	2386
Proximal Policy Optimization (256×256)	28.68 ± 4.32	26.0	34	1052.90 ± 37.21	3826

(Values are mean ± standard deviation.)

C. *Impact of Increasing the Training Budget*

To quantify the effect of additional interactions, Table III reports the difference between the 100,000-step and 50,000-step results. The largest gain is achieved by Proximal Policy Optimization (128×128) with Δ score = +13.10, indicating strong scaling with additional data and improved sample efficiency at smaller capacity. Deep Q-Network (128×128) improves marginally (Δ score = +1.90), while Proximal Policy Optimization (256×256) shows negligible change (Δ score ≈ −0.04), consistent with near-saturation. The tuned Deep Q-Network (256×256) decreases in mean score (Δ score = −0.90) and remains far below the other configurations.

Across models, episodic return and episode length generally increase when the agent learns to survive longer; however, since the return includes shaping and living reward components, it may not be perfectly aligned with the task-success score. Therefore, score provides the most direct and comparable indicator of success for the squash task. The budget sensitivity is visualized in Fig. 3, which summarizes the mean score shift from 50k to 100k steps with ±1 standard deviation.

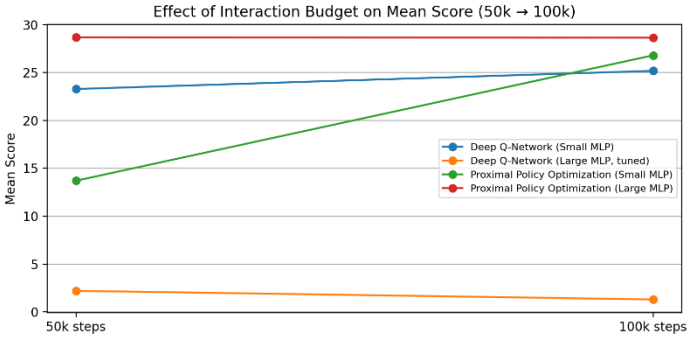

Fig. 4. Effect of interaction budget on mean evaluation score (50 episodes; deterministic policy). Error bars indicate ±1 standard deviation.

TABLE II. *Evaluation summary after 100,000 training steps (N = 50).*

Model	Score (mean ± std)	Median score	Best score	Return (mean ± std)	Mean length
Deep Q-Network (128×128)	25.18 ± 2.63	23.0	30	1250.55 ± 47.55	3607
Deep Q-Network (256×256, tuned)	1.30 ± 1.22	1.0	3	145.58 ± 112.65	481
Proximal Policy Optimization (128×128)	26.80 ± 5.70	22.0	34	909.76 ± 147.71	3680
Proximal Policy Optimization (256×256)	28.64 ± 3.65	28.5	34	918.31 ± 135.62	3832

(Values are mean ± standard deviation.)

D. *Qualitative Trends from Figures*

To complement the tabular summaries, we provide training and evaluation curves. **Fig. 1(a)–1(b)** show the training learning curves (episode return with a moving average) for the 50,000-step and 100,000-step budgets, respectively. **Fig. 2(a)–2(b)** show episode-level evaluation scores across the 50 evaluation episodes for both budgets. These plots highlight two key trends: (i) the larger Proximal Policy Optimization configuration exhibits stable performance with relatively low episode-to-episode variability, and (ii) the smaller Proximal Policy Optimization configuration benefits substantially from the increased budget, consistent with the improvements reported in **Table III**.

Model	Mean score (50k)	Mean score (100k)	Δ score	Mean return (50k)	Mean return (100k)	Δ return	Mean length (50k)	Mean length (100k)	Δ length
Proximal Policy Optimization (128×128)	13.70	26.80	+13.10	495.70	909.76	+414.06	2386.14	3679.56	+1293.42
Deep Q-Network (128×128)	23.28	25.18	+1.90	1339.46	1250.55	−88.91	3463.16	3606.56	+143.40
Proximal Policy Optimization (256×256)	28.68	28.64	−0.04	1052.90	918.31	−134.59	3826.48	3831.66	+5.18
Deep Q-Network (256×256, tuned)	2.20	1.30	−0.90	198.65	145.58	−53.07	679.88	480.66	−199.22

TABLE III. *Performance change when increasing the budget (100,000 − 50,000).*

V. Conclusion and Discussion

This work compared an off-policy value-based method (Deep Q-Network) and an on-policy policy-gradient method (Proximal Policy Optimization) under fixed interaction budgets in a custom Pygame-based squash environment with a Gymnasium-compatible interface. We evaluated two network capacities (128×128 and 256×256 multilayer perceptrons) under two training budgets (50,000 and 100,000 environment steps) and measured performance using deterministic evaluation over N = 50 episodes. Task success was primarily assessed using score, while episodic return and episode length were reported as supportive metrics.

Across both budgets, the larger Proximal Policy Optimization configuration achieved the highest and most stable scores, showing strong performance already at 50,000 steps (Table I) and maintaining a similar level at 100,000 steps (Table II). This suggests that, for this environment, the larger on-policy configuration reaches a near-saturated policy within the tested budget range. In contrast, the smaller Proximal Policy Optimization model benefited substantially from additional interactions, exhibiting the largest improvement when increasing the training budget (Table III). This behavior indicates that limited-capacity on-policy policies can scale effectively with more environment data in this task.

For the value-based approach, the smaller Deep Q-Network improved only modestly with the increased budget, while the tuned larger Deep Q-Network remained substantially weaker across both budgets (Table I–II) and did not benefit from additional interaction steps (Table III). These results highlight the sensitivity of value-based learning to stability and exploration choices, especially under constrained interaction budgets and in environments where dense shaping terms influence the learning signal.

Because the reward includes both shaping and a living reward, episodic return is not perfectly aligned with score. Therefore, score provides the most direct indicator of task success, whereas episodic return should be interpreted as a complementary signal influenced by the reward design. The evaluation protocol with deterministic action selection further ensures that reported performance reflects the learned policy rather than exploration noise.

Limitations and future work. First, each configuration was trained with a single seed; repeating experiments across multiple independent seeds and reporting confidence intervals would strengthen statistical reliability. Second, future work can study the sensitivity of learning to reward shaping by varying shaping strength or using a sparser reward to better align episodic return with task success. Third, the off-policy baseline could be strengthened by incorporating established improvements (e.g., Double Q-learning or prioritized replay) and by performing a broader hyperparameter search. Finally, adding controlled stochasticity to initial conditions (e.g., randomized ball positions and velocities) would provide a clearer assessment of generalization.

References

[1] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," arXiv preprint arXiv:2103.01955, 2021.

[2] R. Kozlica, S. Wegenkittl, and S. Hirländer, "Deep Q-Learning versus Proximal Policy Optimization: Performance Comparison in a Material Sorting Task," arXiv preprint arXiv:2306.01451, 2023.

[3] A. Chen, S. Dewan, et al., "The Use of Reinforcement Learning in Gaming: The Breakout Game Case Study," TechRxiv, 2020, doi: 10.36227/techrxiv.12061728.v1.

[4] N. de la Fuente and D. A. Vidal Guerra, "A Comparative Study of Deep Reinforcement Learning Models: DQN vs PPO vs A2C," arXiv preprint arXiv:2407.14151, 2024.

[5] C. Tan, "Comparative Study of Reinforcement Learning Performance Based on PPO and DQN Algorithms," in Proceedings of CONF-CDS 2025 Symposium: Application of Machine Learning in Engineering, 2025, doi: 10.54254/2755-2721/2025.AST24879.

[6] R. F. Prudencio, M. R. O. A. Máximo, and E. L. Colombini, "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems," arXiv preprint arXiv:2203.01387, 2022.

[7] A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup, "Fast reinforcement learning with generalized policy updates," Proc. Natl. Acad. Sci. U.S.A., vol. 117, no. 48, pp. 30079–30087, 2020, doi: 10.1073/pnas.1907370117.